Useful Scripts

Make Request

Sometimes we might want to make a request to a website programmatically. Instead of having to visit the page in the browser. In Python we can to it the following way.

If you don't have the module requests installed you can install it like this.

`pip install requests`

```
import requests

req = requests.get("http://site.com")
print req.status_code
print req.text
```

Custom headers

We might receive a `403` error if we don't include a user-agent. Or we might want to send a specific header. We can do that the following way.

```
import requests

headers = {
"Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8",
"Accept-Encoding": "gzip, deflate, sdch",
"Accept-Language": "en-US,en;q=0.8,es;q=0.6,sv;q=0.4",
"Cache-Control": "max-age=0",
"Connection": "keep-alive",
"Cookie": "_gauges_unique_hour=1; _gauges_unique_day=1; _gauges_unique_month=1; _gauges_unique_year=1; _gauges_unique=1",
"Host": "docs.python-requests.org",
"If-Modified-Since": "Wed, 03 Aug 2016 20:05:34 GMT",
"If-None-Match": 'W/"57a24e8e-e1f3"',
"Referer": "https://www.google.com/",
"Upgrade-Insecure-Requests": "1",
"User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.82 Safari/537.36"
}

req = requests.get("http://site.com", headers=headers)
print req.status_code
print req.text
```

If you need to add an action, like loggin in or something like that, to your request you do the following:

```
values = {'action' : 'whatever'}
req = requests.get("http://site.com", data=values, headers=headers)
```

Here is the documentation
http://docs.python-requests.org/en/master/user/quickstart/

Read and write to files

Many times we want to read through files and do stuff do it. This can of course be done using bash but we can also do it in python. It might be easier to parse text in python.

```
file_open = open("readme.txt", "r")
for line in file_open:
    print line.strip("\n")
    if line.strip("\n") == "rad 4":
        print "last line"
```

Send requests to your proxy (like Burp)

```
import os
os.environ['HTTPS_PROXY'] = '<proxyurl>:<port>'
# http://127.0.0.1:8080 if it is burp
# Then you need to add verify=False
requests.get("https://google.com", headers=headers,verify=False)
```

Basic banner-grabber

Here is an example of the most basic usage of the socket module. It connects to a port and prints out the response.

```
#!/user/bin/env python

# Importing the socket module
import socket

# We use the socker() method of the module socket and store it in the variable s.
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Here we use the connect method of the socket we created. The two arguments are pretty self-explanatory
# The first is the adress the second is the port.
s.connect(("192.168.1.104", 22))

# Here we save what the socket reviewed in the variable answer.
answer = s.recv(1024)
print answer

# Send stuff. REMEMBER THE \r\n

s.send("this is my message\r\n")
print s.recv(1024)

# Here we close the socket.
s.close
```

If you need to check all 65535 ports this might take some time. If a packet is sent and recieved that makes it 65535 seconds, it translates into about 18 hours. So to solve that we can run the a function in new threads.

```
from multiprocessing.dummy import Pool as ThreadPool
pool = ThreadPool(300)
results = pool.map(function, array)
```

Read more about parallellism here: http://chriskiehl.com/article/parallelism-in-one-line/

Connecting to SMTP

A crappy script to connect to a smtp-server and if you are allowed to test for users with VRFY it goes ahead and test for the users that you input from a file.
One very important thing to note here, that had me stuck for quite a while is that you need to send the query strings in raw-format

The `\r` here is fundamental!!

```
s.send('VRFY root \r\n')
```

```
#!/usr/bin/python
import socket
import sys
import time
import re

ips = [
"192.168.1.22",
"192.168.1.72"
]

users = ["root"]

userfile = open("/fileWithUsernames.txt", "r")
for line in userfile:
    user = line.strip("\n")
    users.append(user)


for ip in ips:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((ip, 25))
    banner = s.recv(1024)

    print "****************************"
    print "Report for " + ip
    print banner
    s.send('VRFY root \r\n')
    answerUsername = s.recv(1024)
    answerAsArray = answerUsername.split(" ")

    if answerAsArray[0] == "502":
        print "VRFY failed"
    if answerAsArray[0] == "250":
        print "VRFY command succeeded.\nProceeding to test usernames"

        for username in users:
            time.sleep(5)
            s.send("VRFY " + username + "\r\n")

            answerUsername = s.recv(1024)
            answerUsernameArray = answerUsername.split(" ")
            print answerUsernameArray[0]
            if answerUsernameArray[0] == "250":
                print "Exists: " + username.strip("\n")
            else :
                print "Does NOT exist: " + username.strip("\n")
    if answerAsArray[0] == "252":
        print "FAILED - Cannot verify user"
    else:
        "Some other error or whatever here it is: \n" + answerUsername



    s.close()
```

Parsing html

Install beautifulsoup 4

```
sudo apt install python-bs4
```

```
import requests
from bs4 import BeautifulSoup
r = requests.get("https://google.com")
soup = BeautifulSoup(r.text, 'html.parser')
print soup.find_all("title")
```

Client/Server using sockets

http://programmers.stackexchange.com/questions/171734/difference-between-a-socket-and-a-port